Model paralelismu – PC gramatické systémy

Radek Kočí

MTI

8. ledna 2002

Abstrakt

Distribuované a paralelní výpočty. To je oblast informatiky, která je v současné době velkým trendem v informatice a dotýká se svým způsobem každé její oblasti. Autor tohoto článku se zabývá modely distribuovaných systémů a hlavně distribuovanými nástroji pro modelování těchto systémů. Každý složitější nástroj se potýká s problémem formálního popisu svých technik, které by umožnily např. formální analýzu jeho chování. Tato práce se pokouší provést čtenáře formálními specifikacemi paralelních modelů, největší část je zaměřena na PC gramatické systémy.

1 Úvod

V současné době je kladen velký důraz na distribuované a paralelní výpočty a s nimi související modely distribuovaných a paralelních výpočtů. Důvodů směřujících k tomuto trendu je několik. My se zmíníme o dvou, které jsou podle názoru autora článku nejpodstatnější – rychlost výpočtu a možnost tvorby multiaplikací, tedy takových aplikací, jejichž jednotlivé části mohou být realizovány v různých prostředích. Jinými slovy, distributivita zjednodušuje realizaci, v naší výzkumné skupině tolikrát skloňovaného, heterogenního prostředí pro modelování a simulaci systémů. V textu tohoto článku se přeneseme od aplikací na abstraktnější úroveň – úroveň modelů distribuovaných a paralelních výpočtů.

Zkusme se nyní zamyslet nad vztahem mezi distributivitou a paralelismem. Distribuovaný výpočet je obecně tvořen množinou navzájem nezavislých, komunikujících jednotek, které provádí sekvenční výpočet (proces)[1]. Pokud abstrahujeme od takových věcí, jako je možnost realizace jednotlivých jednotek na různých platformách, různých místech sítě, časová zpoždění komunikací apod. (což my v tomto článku předpokládáme), můžeme distribuovaný výpočet považovat za množinu navzájem nezávislých *paralelních* procesů, které spolu mohou komunikovat prostřednictvím tzv. *synchronizačních bodů*. V tomto okamžiku

[1] Proces považujeme za sekvenční výpočet.

můžeme rozšířit definici jednotky z jednoho procesu na množinu *paralelních* procesů bez újmy na obecnost. V dalším textu se tedy budeme zabývat modely paralelních komunikujících procesů.

Převážná část práce je věnována gramatickým systémům, konkrétně PC gramatickému systému. Zabývá se formálním popisem základní struktury a vlastností PC systému. Dále si všímá rozšíření komunikačního mechanismu PC systému. V následující části je pak (pouze informativně) proveden nástin jiných možností formálního popisu komunikace distribuovaných procesů než pomocí gramatiky. V závěru článku se pak autor zamýšlí nad možnostmi využití těchto formálních popisů.

2 Gramatické systémy

Klasická teorie formálních jazyků a automatů modeluje klasické *centralizované* výpočty, kde je výpočet uskutečňován jedním centrálním *agentem*. To odpovídá tomu, že jazyk je generován jednou gramatikou nebo přijímán jedním automatem. V moderním pojetí distribuovaného výpočtu však toto nestačí. Proto byl vytvořen gramatický systém jako gramatický model pro distribuované výpočty, který definuje a analyzuje pojmy jako jsou distribuce, paralelismus, souběžnost a komunikace. Gramatický systém je množina gramatik, pracujících souběžně podle specifikovaného protokolu za účelem generování jednoho jazyka. Rozlišujeme dvě základní třídy gramatických systémů – *sekvenční* a *paralelní*. My se zaměříme na paralelní gramatické systémy.

PCGC (*Parallel Communicating Grammar System*) je tvořen komponentami, které pracují s vlastní větnou formou. Všechny komponentní gramatiky mají společný čas. V každém časovém kroku každá komponenta aplikuje pravidlo, které přepíše její větnou formu. Klíčovou vlastností PCGC je komunikační mechanismus založený na tzv. *dotazech*. Zkusme si jej nyní neformálně popsat. Označme si každou komponentu indexem i, $i \in \langle 1, n \rangle$, kde n je počet komponent. Každé pravidlo komponent může generovat speciální symbol Q_i. Pokud komponenta i generuje speciální symbol Q_j, pak aktuální větná forma komponenty j bude poslána komponentě i, kde přepíše ve větné formě symbol Q_j. Jedna z komponent je navržena jako *master* a jazykem generovaným gramatickým systémem se pak rozumí jazyk generovaný touto *master* komponentou.

Než se na tento systém podíváme podrobněji, pokusme si odpovědět na otázku - proč gramatické systémy? V současné počítačové vědě, umělé inteligenci a jiných obdobných směrech se zabýváme komplexními úlohami, které jsou distribuovány prostřednictvím agentů (procesorů), pracujících současně, s jasně definovaných chováním. Praktickou formu těchto myšlenek můžeme spatřovat v paralelních výpočtech, počítačových sítí apod. Teorie formálních jazyků má dobré prosředky pro modelování aspektů systému, v kontextu výše zmíněných systémů však vyvstává otázka, jak popsat chování multi-agentních systémů v termínech jazyka, který popisuje daný systém. Gramatický systém se pokouší tuto otázku zodpovědět. Komponenty systému mohou korespondovat s agenty, aktuální generované řetězce korespondují se symbolickým prostředím a chování systému je reprezentováno jazykem nebo sekvencemi řetězců, které identifikují aktuální stav. Můžeme si představit, že PC gramatické systémy studují sítě s komponentami, které navzájem komunikují prostřednictvím zasílaných řetězců.

2.1 Formální notace

Nyní si formálně zavedeme několik základních pojmů a definic. Pro abecedu V, délka řetězce $x \in V^*$ je zapisována jako $|x|$. Pokud $W \subseteq V$, pak $|x|_W$ reprezentuje počet výskytů znaků z abecedy W v řetězci x.

Chomského gramatika je definována jako

$$G = (N,\, T,\, S,\, P)$$

kde N je abeceda nonterminálů, T je abeceda terminálů, $S \in N$ je počáteční symbol (axiom) a P je množina přepisovacích pravidel tvaru $u \to v,\ u, v \in (N \cup T)^*,\ |u|_N \geq 1$. Přímá derivace je definována jako $x \Longrightarrow y\, iff\; x = x_1 u x_2$a $y = x_1 v x_2$ pro nějaké $u \to v \in P$.

Symbolem $\Longrightarrow^*$ označme tranzitivní uzávěr relace $\Longrightarrow$. Jazyk generovaný gramatikou G je pak definovaný jako

$$L\left(G\right) = \left\{x \in T^* |\, S \Longrightarrow^* x\right\}.$$

Označme si symboly REG, LIN, CF, CS a RE rodiny regulárních, lineárních, bezkontextových, kontextových a rekurzivně vyčíslitelných jazyků.

2.2 PC gramatické systémy

Nyní si definujme PCGS formálněji. PC gramatický systém stupně n, $n \geq 1$, je $(n+3)-$ *tice*

$$\Gamma = \left(N,\, K,\, T,\, \left(S_1,\, P_1\right),\, \ldots,\, \left(S_n,\, P_n\right)\right),$$

kde N a T jsou abecedy nonterminálů a terminálů, $K = \{Q_1,\, Q_2,\, \ldots,\, Q_n\}$ je množina speciálních symbolů, P_i je konečná množina přepisovacích pravidel nad $N \cup K \cup T$ a $S_i \in N$ pro každé $1 \leq i \leq n$.

Množina P_i, $1 \leq i \leq n$ se nazývá *komponenty* gramatického systému a elementy $Q_1,\, \ldots,\, Q_n \in K$ se nazývají *dotazovací symboly*, index i koresponduje s i-tou komponentou P_i. Na komponenty gramatického systému Γ můžeme také nahlížet jako na nezávislé gramatiky. Pak můžeme systém Γ popsat následujícím způsobem: $\Gamma = (N,\, K,\, T,\, G_1,\, \ldots,\, G_n)$, kde $G_i = \left(N \cup K,\, T,\, S,\, P_i\right),\, 1 \leq i \leq n$.

Pravidla derivace: Uvažujme PC gramatický systém $\Gamma = \left(N,\, K,\, T,\, \left(S_1,\, P_1\right),\, \ldots,\, \left(S_n,\, P_n\right)\right)$ stupně n a dvě n-tice $(x_1,\, x_2,\, \ldots,\, x_n)$ a $(y_1,\, y_2,\, \ldots,\, y_n)$, kde $x_i,\, y_i \in V_\Gamma^*$, $1 \leq i \leq n$ a $x_1 \notin T^*$, píšeme $(x_1,\, x_2,\, \ldots,\, x_n) \Longrightarrow (y_1,\, y_2,\, \ldots,\, y_n)$ v jednom z následujících případů

- pro každé i, $1 \leq i \leq n$, $|x_i|_K = 0$ a pro každé takové i, existuje buď $x_i \Longrightarrow y_i$ podle pravidla P_i, nebo $x_i = y_i \in T^*$
- existuje i, $1 \leq i \leq n$, $|x_i|_K > 0$. Uvažujme pro každé i: $x_i = z_1 Q_{i_1} \ldots z_t Q_{i_t} z_{t+1}$, $t > 1$ pro $z_j \in (N \cup T^*)$, $1 \leq i \leq t+1$. Jestliže pro každé j, $1 \leq j \leq t$ platí $\left|x_{i_j}\right|_K = 0$, potom $y_i = z_1 x_{i_1} \ldots z_t x_{i_t} z_{t+1}$ a $y_{i_j} = S_{i_j}$. Jestliže pro nějaké j, $1 \leq j \leq t$, $\left|x_{i_j}\right|_K \neq 0$, potom $y_i = x_i$. Pro všechna i, $1 \leq i \leq n$ taková, že y_i nebylo výše specifikováno, platí $y_i = x_i$.

N-tici $(x_1, \ldots, x_n)$, kde $x_i \in V_\Gamma^*$ pro každé i, $1 \leq i \leq n$, nazýváme *konfigurace* systému Γ.

Zkusme si nyní osvětlit pravidla derivace neformálně. Předpokládejme výchozí konfiguraci $(x_1, \ldots, x_n)$ a cílovou konfiguraci $(y_1, \ldots, y_n)$ jednoho kroku derivace PC gramatického systému. Mohou nastat tyto případy:

- ve všech větných formách $x_1, \ldots, x_n$ není obsažen dotazovací symbol Q_i a máme derivaci $x_i \Rightarrow y_i$ pro každou komponentu P_i. Jestliže pro nějakou větnou formu x_i platí $x_i \in T^*$, potom $y_i = x_i$.
- v nějaké (nejméně jedné) větné formě x_i je obsažen dotazovací symbol Q_j. V tomto případě dochází ke *komunikačnímu kroku*, který má vždy přednost před derivací podle pravidel jednotlivých komponent P_i, $1 \leq i \leq n$. Komunikační krok probíhá následujícím způsobem: každý výskyt symbolu Q_j je přepsán větnou formou x_j, pokud tento řetězec neobsahuje dotazovací symbol. Jinými slovy, v jednom komunikačním kroku komponenty PCGC je dovoleno přepsat pouze *jeden* dotazovací symbol. Pokud nějaký dotazovací symbol nemohl být přepsán v tomto komunikačním kroku, může být přepsán v kroku následujícím.

Deadlock PC gramatický systém může být při derivaci zablokován v těchto dvou případech:

- V žádné větné formě x_i není dotazovací symbol, větná forma x_i aktuální konfigurace $(x_1, \ldots, x_n)$ není terminální řetězec a neexistuje pravidlo komponenty P_i, které by mohlo být aplikováno na řetězec x_i.
- Cyklické odkazování dotazovacích symbolů. Pokud P_{i_1} generuje Q_{i_2}, P_{i_2} generuje Q_{i_3} a tak dále až po $P_{i_{k-1}}$ generuje Q_{i_k} a P_{i_k} generuje Q_{i_1}. V tomto případě není možná regulární derivace podle pravidel P, protože komunikace má vyšší prioritu, a současně nelze provést přepis na základě komunikace, protože pouze řetězec, který *neobsahuje* dotazovací symbol může prostřednictvím komunikace nahradit příslušný dotazovací symbol.

Jazyk generovaný PC gramatickým systémem Γ

$$L\left(\Gamma\right)=\left\{x\in T^{*}\mid\left(S_{1},\,S_{2},\,\ldots,\,S_{n}\right)\Rightarrow^{*}\left(x,\,\alpha_{2},\,\alpha_{3},\,\ldots,\,\alpha_{n}\right),\,\alpha_{i}\in V_{\Gamma}^{*},\,2\leq i\leq n\right\}.$$

Derivace začíná n-ticí $(S_1,\, S_2,\, \ldots,\, S_n)$ a pokračuje přepisovacími a komunikačními kroky. Derivace končí v okamžiku, kdy komponenta P_1 vygeneruje terminální řetězec, přičemž nezáleží na obsahu řetězců ostatních komponent. Komponenta P_1 se nazývá *master* gramatického systému.

Na základě komunikace mezi komponentami rozlišujeme dvě základní třídy PC gramatických systémů. Předpokládejme systém $\Gamma = (N,\, K,\, T,\, (S_1,\, P_1)\,,\, \ldots,\, (S_n,\, P_n))$. Pokud pouze P_1 může generovat dotazovací symbol (formálně $P_i \subseteq (N \cup T^*) \times (N \cup T^*)$ *pro* $2 \leq i \leq n$), potom říkáme, že Γ je centralizovaný PC gramatický systém; v jiném případě Γ je decentralizovaný systém.

PC systém nazýváme *returning*, pokud po komunikaci každá komponenta, která zaslala svůj řetězec jiné (dotazovací) komponentě, vrací *axiom* (svůj prvotní nonterminální symbol). PC systém nazýváme *non-returning*, pokud po komunikaci každá komponenta, která zaslala svůj řetězec jiné (dotazovací) komponentě, *axiom* nevrací, ale pokračuje dále nad svým aktuálním řetězcem.

2.3 Rozšíření PC gramatických systémů

Dosud jsme si popsali PC gramatický systém, jehož jednotlivé komponenty pracují synchronně (tj. existuje jakési fiktivní hodiny, které jsou pro každou komponentu sdílené) a mohou spolu komunikovat (vznášet požadavky – dotazy) prostřednictvím speciálních symbolů. Nyní se zaměříme na rozšíření komunikačního mechanismu tohoto systému.

Z praktického hlediska je někdy vhodné rozšířit komunikační mechanismus z výše uvedého typu (tj. vznášení požadavků) na komunikaci pomocí příkazů, což může např. odpovídat komponentám sítě, které zasílají zprávy. Základní myšlenka je stejná jako u obvyklých PC gramatických systémů – každá gramatika pracuje separátně nad svou větnou formou (řeší svůj podproblém). Navíc má každá gramatika svůj jazyk, který má úlohu vstupního filtru. V určitých situacích je přerušeno přepisování větných forem a každá komponenta zašle svou aktuální větnou formu ostatním komponentám, pro které zasílaná větná forma odpovídá vstupnímu filtru.

Tento nový přístup v komunikaci s sebou nese otázky, které je potřeba zodpovědět před vlastní formulací modelu:

- Kdy přerušit přepisování řetězců a provést komunikační krok? Dva možné přístupy jsou provést komunikaci po každé derivaci v každé komponentě synchronně, nebo po maximálním možném počtu derivací v každé komponentě.

- Která část řetězce se účastní komunikace – celý řetězec nebo nějaká jeho část?

- Provádět komunikaci v *returning* nebo *non-returning* módu?

- Jak řešit situaci, kdy více zpráv (řetězců) komunikuje se stejnou komponentou?

Podívejme se nyní na jednu z možných variant tohoto typu rozšíření PC gramatického systému. Předpokládejme, že systém provádí komunikační krok až poté, co došlo k maximálnímu počtu aplikací přepisovacích pravidel všech komponent. Na komunikaci se podílí celý řetězec, který přepisuje řetězec cílové komponenty a provádí konkatenaci zaslaných řetězců v jedné komponentě. Nyní můžeme tento systém formálně zapsat:

$$\Gamma = (N,\, K,\, T,\, (S_1,\, P_1,\, R_1)\,,\, \ldots,\, (S_n,\, P_n,\, R_n))\,,\, n \geq 1,$$

kde vidíme rozšíření komponent oproti původní definici o jazyk, definující vstupní filtr R_i, $1 \leq i \leq n$, $R_i \subseteq N \cup T^*$.

Aplikaci systému si ukážeme na jednoduchém příkladu. Předpokládejme následující gramatický systém:

$$\Gamma = (N,\, \{a,\, b,\, c\}\,,\, (S_1,\, P_1,\, R_1)\,,\, (S_2,\, P_2,\, R_2)\,,\, (S_3,\, P_3,\, R_3))\,,$$

$N = \left\{S_1,\, S_2,\, S_2',\, S_3,\, S_3',\, X\right\}$,
$P_1 = \{S_1 \to aS_1,\, S_1 \to bS_1,\, S_1 \to X\}$, $R_1 = \{a,b\}^*\, c$,
$P_1 = \left\{S_2 \to S_2',\, X \to c\right\}$, $R_2 = \{a,b\}^*\, X$,
$P_1 = \left\{S_3 \to S_3',\, X \to c\right\}$, $R_3 = \{a,b\}^*\, X$.

Kroky derivace:
$(S_1,\, S_2,\, S_3) \Rightarrow \left(xX,\, S_2',\, S_3'\right)$ pro nějaké $x \in \{a,\, b\}^*$.
V dalším kroku následuje komunikace (nelze provést žádné přepisovací pravidlo), tj. větné formy jednotlivých komponent se zašlou ostatním a porovnají se se vstupním filtrem. Např. forma xX první komponenty odpovídá jazykům R_2 a R_3, takže se "pošle" odpovídajícím komponentám, kde nahradí stávající řetězec a v první komponentě se nahradí jejím počátečním symbolem:
$\left(xX,\, S_2',\, S_3'\right) \vdash (S_1,\, xX,\, xX)$.
Další aplikace přepisovacích pravidel a komunikací:
$(S_1,\, xX,\, xX) \Rightarrow (yX,\, xc,\, xc) \vdash (xcxc,\, yX,\, yX) \Rightarrow (xcxc,\, yc,\, yc)$ pro nějaké $y \in \{a,\, b\}^*$.

2.4 Jiný pohled na PC gramatické systémy

Nyní se trošku zmíníme o některých pojmech, které nám umožní nahlédnout na význam modelů paralelních a distribuovaných výpočtů. Jedním z nich je pojem *blackboard model*. Jde o abstraktní model výpočtu, který obsahuje tyto hlavní součásti: *blackboard* (sdílená datová struktura, obsahující aktuální stav řešeného problému), tzv. znalostní zdroje (agenti, procedury, procesory apod.), které se podílejí na řešení problému změnami obsahu *blackboard*u a protokol pro řízení kooperace mezi zdroji. Mírnou modifikací této představy získáme tzv. *classroom model*, který je podobný PC gramatickému systému. Tento model se vyznačuje těmito vlastnostmi:

- každý agent (znalostní zdroj, procesor, gramatika) má vlastní "tabuli", která obsahuje popis podproblému, který řeší daný agent
- každý agent operuje pouze v rámci své "tabule".
- existuje jeden agent, který operuje nad *blackboard*em, tento agent zná popis celého problému. Bývá označován jako *team-leader.*
- agenti komunikují prostřednictvím požadavků na jejich "tabule"

Pokud si nyní představíme PC gramatický systém jako typ classroom modelu, můžeme říci, že *master* gramatika plní úlohu *team-leader*u a ostatní gramatiku úlohu *team-procesů.* U centralizovaného PC systému je tou pouze *leader*, kdo může vznášet požadavky na komunikaci - ostatní pouze slepě pracují nad svým podproblémem.

3 Jiné paralelní modely

Jak jsme viděli v předchozí kapitole, PC gramatický systém umí formálně popsat paralelní komunikující výpočty prostřednictvím gramatik a pravidel pro vzájemnou kooperaci. Současně existuje matematický aparát pro analýzu takto definovaných systémů. Zkusme se nyní podívat na modely paralelních a distribuovaných výpočtů, které nejsou založeny na gramatice.

3.1 Cellular automata

Cellular Automata (CA) byly poprvé představeny pány Ulamem a von Neumannem v letech 1940. CA představují formální rámec pro popis chování komplexních dynamických systémů. CA je dynamický systém s diskrétním prostorem a časem. CA je obvykle specifikován mříží (*lattice*) L buněk (*cells*), konečnou množinou stavů Q pro buňky, konečným *prostředím* (*neighbourhood*) N a lokální přechodovou funkcí $\delta : Q^N \to Q$.

Míříž buněk může být n-dimenzionální, nejčastěji používaný počet dimezní je $n = 1, 2, 3$. Každá z buněk může být v jednom z konečného počtu stavů, definovaných množinou Q. Můžeme tedy říci, že každá buňka reprezentuje konečný automat s rozšířenou přechodovou funkcí. Ta je definována prostřednictvím prostředí N a vnáší do ní okolí aktuální buňky (konečného automatu), tj. stavy okolních buněk.

Buňky v mřiži L tedy reprezentují stavy jednotlivých konečných automatů a následující stav je ovlivněn stavy okolních buněk. Je definován tzv. *rádius* r, který určuje velikost okolí dané buňky, tedy počet buněk, kterými je tato buňka ovlivňována.

Jak je vidět z předchozího popisu, CA definují pro každý konečný automat stejnou přechodovou funkci (reprezentovanou prostředím N), každý z konečných automatů má stejnou množinu stavů a každý automat může komunikovat (ve smyslu být ovlivněn) pouze s automaty *ve svém okolí.* Takto definovaný systém jistě neodpovídá našim požadavkům na model paralelního systému. Nicméně autor považoval za zajímavé uvést tento typ automatu, neboť se s ním pojí řada jiných typů automatů v rámci výzkumu, který probíhá na *University of Karlsruhe*. Jedním ze zajímavých typů je tzv. *parallel pointer machine* (PPM). PPM se skládá z konečného počtu konečných automatů. Každý automat má jeden výstup, ze kterého je možné číst aktuální stav automatu a fixní počet vstupů, které se napojeny na výstupy jiných konečných automatů. V každém časovém kroku každý konečný automat provede synchronně akci podle pravidel, které mohou záviset na aktuálním stavu automatu a aktuálních vstupech. Akce může představovat překlopení do nového stavu (tedy provedení přechodu v klasickém konečném automatu), specifikace vytvoření nového konečného automatu nebo specifikace přepojení některých vstupů na výstup jiného automatu.

4 Závěr

V počítačovém světě dnešní doby se velice prosazují systémy, které jsou označovány jako distribuované a paralelní. S nimi ovšem vyvstala i snaha o formální popis těchto systémů, který by umožnil formální analýzu chování systémů. Cílem tohoto mini-článku bylo zamyslet se právě nad možnostmi formálních modelů distribuovaných paralelních systémů. Základní rovinu textu tvoří gramatické systémy, konkrétně PC gramatické systémy. Tyto systémy vznikly jako reakce na požadavek formálního popisu komplexních distribuovaných systémů. Existuje celá řada typů gramatických systémů, které prošly od svého vzniku. Např. v roce 1988 byly představeny *cooperating/distributed grammar systems* pro modelování syntaktických aspektů řešení problémů na základě *blackboard model*u. PC gramatické systémy studují sítě sestavené z komponent, které spolu komunikují datovými řetězci prostřednictvím požadavků. Každou komponentu popisují gramatikou a definují přesná pravidla komunikačního mechanismu mezi jednotlivými komponentami. Gramatické systémy ovšem nejsou jedinou cestou pro formální modelování paralelních systémů. Jako příklad jiným cest byly v článku uvedeny celulární automaty. Tady se musí autor zastavit a s lítostí konstatovat, že nevěnoval dostatečnou pozornost jiným způsobům formálního popisu paralelních systémů než na základě gramatiky. Závěrem lze říci, že existuje celá řada výzkumů v této oblasti, přičemž lze vysledovat dva protichůdné požadavky: dostatečná formalizace z důvodů dobré analyzovatelnosti systémů a dostatečná "jednoduchost" vytvoření formálního modelu.